Lossless Coding Standards for Space Data Systems

Robert F. Rice
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Abstract

The International Consultative Committee for Space Data Systems (CCSDS) is preparing to issue its first recommendation for a digital data compression standard. Because the space data systems of primary interest are employed to support scientific investigations requiring accurate representation, this initial standard will be restricted to lossless compression.

In particular, the demonstrated breadth of applicability and high performance of what has become known as "The Rice Algorithms", has led the CCSDS to recommend a version of these algorithms. The key algorithmic and performance characteristics that have driven the development of this recommendation are discussed.

1. Introduction

The Consultative Committee for Space Data Systems (CCSDS) is an international group dedicated to providing sound technical solutions that are common problems to all participants. Member space agencies include the National Aeronautics and Space Administration, the European Space Agency and agencies representing the United Kingdom, Canada, Russia, France, Germany, Brazil and Japan. In addition, there are 22 international observer agencies as well. The committee has previously issued recommendations for the standardization of space-based telemetry and channel coding systems. They are currently preparing such a recommendation for a digital data compression standard.[1]

The space data systems of interest are primarily employed to support scientific investigations which have an inherent demand for accurate data representation. Consequently, this first issue of a data compression standard will be restricted to lossless compression.

The "Rice Algorithms" generally refer to a collection of easily implemented adaptive techniques for lossless data compression. These techniques have already been widely used, in various forms, to support a broad range of space data acquisition and storage problems. This demonstrated breadth of applicability and high performance when applied to real problems led the CCSDS to recommend a version of the Rice Algorithms for their initial data compression standard. The key algorithmic and performance characteristics that have driven the development of this recommendation are discussed here.

Additional tutorial information and a broader historical perspective on related developments for space communication can be found in Ref. 2. Precise details of the standard recommendation are not provided here. The reader is referred to Ref. 1 for the definitive specification.

Many details of this paper, including style and notation, are drawn from Ref. 3, which can be considered the primary reference. The coding principals, along with notation, are developed here in a step-by-step fashion. Thus, it may be difficult to derive full comprehension when skipping over intervening material. It is a goal of this paper that a reader will conclude: "I understand why they pursued this standard."

11. The overall Coding Problem

As shown in Fig. 1, the lossless coding process can be partitioned into two steps:

Step 1) Reversible preprocessing of a block of data samples $\vec{x}$ into another data block δ that has certain "standard" characteristics, and

Step 2) Using adaptive variable-length coding to efficiently represent the Standard Source $\tilde{\delta}$ block produced by Step 1.

The reversible preprocessing step is crucial to the overall coding problem but it is secondary to the main thrust of this paper, which will focus on Step 2.

The Standard Source

The output of the idealized preprocessor and input to the adaptive variable length coder are blocks of data, $\tilde{\delta}$, that exhibit certain characteristics. First, let

$$\delta = \delta_1 \delta_2 \ldots \delta_J$$

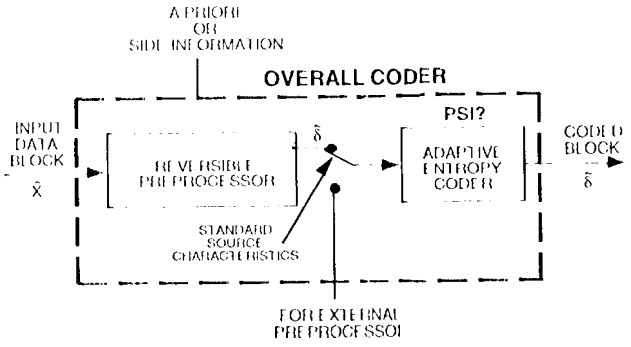

Fig. 1. The Two Steps

be such a J sample block. Then

A) The J samples are values from the non-negative integers

B) The samples of $\tilde{\delta}$ have the probability distribution

$$P_\delta = \{p_0, p_1, \ldots\}$$

C) The $\{p_i\}$ exhibit the ordering $p_0 \geq p_1 \geq p_2 \geq$

D) The samples of $\tilde{\delta}$, are independent with themselves and any available a priori or side information.

While idealized, these conditions can be well approximated for many practical problems. In any case, it is the preprocessor's task to achieve and maintain these conditions as closely as possible. Consider the consequence for the second step in Fig. 2.

The coder always has to deal with the same alphabet (with the exception of its size) regardless of the originating source. The proper assignment of shorter codewords to the most likely integers is established for any variable length code by the probability ordering in (C). The condition in (D) means that the burden of making the most from data correlation and a priori knowledge is placed on the preprocessor. If correlation still exists in $\tilde{\delta}$, then perhaps the preprocessor can be improved, yielding $\tilde{\delta}$ distributions in which the smaller integers occur more frequently.

Coder Performance

The entropy of a particular distribution P_δ is given by

$$I(P_\delta) = H_\delta = -\sum_j p_j \log_2 p_j \qquad (1$$

If P_δ is unchanging, then H_δ is a bound to the best performance of any coding algorithm that follows. Assuming that condition (D) has been met, H_δ is also a bound to the overall performance in Fig. 1. For a given P_δ, the best single variable length code can be derived from the Huffman algorithm.[4] But the real world hardly ever produces P_δ which don't change. ***The real practical problem facing the coder of preprocessed $\tilde{\delta}$ blocks is to maintain efficient performance as P_δ changes.***

We say that a coder is efficient if it obtains performance "close to" an average measured entropy, $\hat{H}_\delta$, which will vary as P_δ does. Here "close to" usually means within about 0.2 to 0.3 bits/sample. For very low entropies a smaller number would be more appropriate.

III. More On The Preprocessor

A Predictive Preprocessor

For many practical problems (e.g., imaging) the simple predictive preprocessor shown in Fig. 2 can be used to establish a good approximation to the desired conditions for a Standard Source.

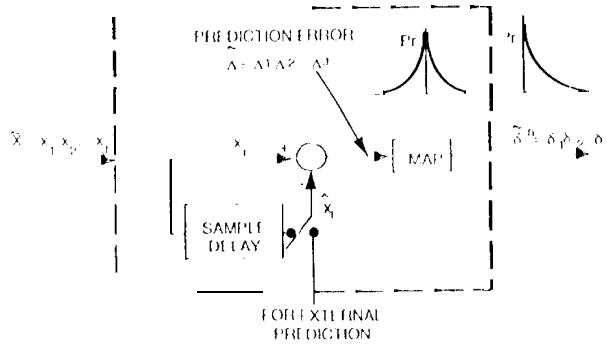

Fig. 2. Basic Preprocessor

The Predictor. The first part of this preprocessor is a very simple one-dimensional predictor that simply predicts that the next sample will be the same as the last. This typically produces unimodal error distributions, often accurately modelled as Laplacian. The J sample input sequence $\tilde{X}$ is converted into a J sample error sequence $\tilde{\Lambda}$ in the figure. For initialization purposes as well as to allow for an alternate external predictor (e.g., two-dimensional) the figure also includes a predictor switch. For these reasons, recent VLSI implementations include this feature

The Mapper. The error distributions generated by such predictors reliably approximate the condition

$$Pr[0] \geq Pr[1] \geq Pr[-1] \geq Pr[2] \geq \ldots \quad (2)$$

Under most conditions it is a simple matter to map these positive and negative errors into the non-negative integers such that condition (C) for a Standard Source is also well approximated - simply implement Table 1 which puts positive errors first or Table 2 which puts negative errors first

Table 1. Basic Map*

Error Value Δ_i	Mapped δ_i
0	0
+1	
-1	2
+2	3
-2	4
+3	5

Table 2. Basic Map"

Error Value Δ_i	Mapped δ_i
0	0
1	1
+1	?
-2	3
+2	4
-3	5

But when diffs values are close to their limits, condition (2) and hence condition (C) cannot be true for all values. For example, if an input sample x_i were zero, any negative prediction error would be impossible (i.e., with zero probability). If $x_i = 1$, then the only negative error possible is -1 and so on.

Modified mappings for Tables 1 and 2 were defined in Ref. 5 to account for this boundary effect and remove the zero probability possibilities from the preprocessor's output. For some test images this netted an overall performance improvement of about .03 bits/sample.

While easily implemented, the modified mappings provided an additional bonus. When input data samples, x_i, are quantized to n bits, so too are the resulting preprocessed samples, δ_i.

A modified mapping was first used in a software image compressor on the Voyager spacecraft.[5] All current implementations utilize one of the two modified mapping functions. Ref. 3 provided the following interesting result: A "decoder" using the modified map for Table 1 can be used to decode data that has been generated using the modified map for Table 2, and vice versa.

Preprocessor Switch

While these sample predictive preprocessors are applicable to a broad set of real problems, more powerful, adaptive or distinctly different forms of preprocessing may be more suitable to some problems. For this reason, recent VLSI implementations of the techniques described herein have included a "preprocessor switch" (see Fig. 1) to allow alternate preprocessors to be combined with the adaptive variable length coding described in later sections.

IV. Adaptive Variable Length Coding

We can now focus entirely on the problem of efficiently coding blocks of preprocessed data that exhibit the characteristics of a Standard Source in A) - D), but which can rapidly change.

Notation, Definitions

The notation conventions described below are consistent with those used in many of the principal references to this work.

Names. The naming of coding algorithms initially subscripted and superscripted the greek letter, ψ (psi). Here, and in some of the later references the greek letter will be replaced by the english "PSI". For example, one of these coders will appear as

PSIi,k

Then, if PSIi,k is applied to a block $\vec{\delta}$, the resulting sequence will be designated

PSIi,k [$\vec{\delta}$]

Dynamic Range. The range of entropies over which a particular coder can be viewed as efficient is called its Dynamic Range.

Concatenation. The asterisk, *, will be used to indicate concatenation when emphasis is needed.

Code Parameters. Some of these will be introduced as they occur. But we will consistently use the following definitions:

N = the number of code options in an adaptive coder

$\tilde{\delta} = \delta_1\ \delta_2 \ldots \delta_J$ is an input block of J preprocessed samples

n = number of bits of quantization for an input sample

$\lceil x \rceil$ = smallest integer greater than or equal to x

$\mathcal{L}(\tilde{S})$ = length of sequence $\tilde{S}$ in bits or samples

The Adaptive Coder

Fig. 3 provides a block diagram of a general N-option adaptive coder named PSI 11 in Ref. 6.

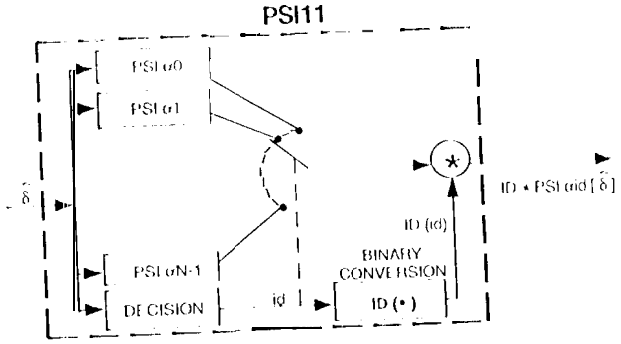

Fig 3 The Adaptive Coder

Here, a bank of N-1 coders labeled $PSI\alpha_0$, $PSI\alpha_1$, ... are shown. The α_j simply represents the label used for the jth coder. For example, if the jth coder were named PSIwow then α_j = wow.

The adaptive coder works as follows. A decision mechanism outputs the number, id, of the code option that would produce the shortest coded sequence when applied to the input block $\tilde{\delta}$. Decisions can be based on a comparison of actual or estimates of coded results for each algorithm. This shortest coded sequence is then

$$PSI\alpha_{id}[\tilde{\delta}] \tag{3}$$

which is passed on to the coder's output and prefixed with a binary version of the identifier, id, denoted as ID, to tell a decoder which coding algorithm was used to represent $\tilde{\delta}$. Thus the output of PSI11 takes the form

$$PSI11[\tilde{\delta}] = ID * PSI\alpha_{id}[\tilde{\delta}] \tag{4}$$

Specific Code Options

We can now fill in the general block diagram for adaptive coder, PSI1, with very specific code options.

Fundamental Sequence Code, PSI1. Define the codeword fs[i] by

i zeroes

The length of "code word" fs[i] is

$$\mathcal{L}(fs[i]) = i + 1 \text{ bits}$$

Applying fs[·] to $\tilde{\delta}$ we get

$$PSI1[\tilde{\delta}] = fs[\delta_1] * fs[\delta_2] * \ldots fs[\delta_J] \tag{6}$$

That is, code option PSI1 is the application of the "fs code" in (5) to all the samples of $\tilde{\delta}$. The resulting coded block has been called the Fundamental Sequence (since about 1970).

The length of a Fundamental Sequence is

$$F = \mathcal{L}(PSI1[\tilde{\delta}]) = J + \sum_{j=1}^{J} \delta_j \tag{7}$$

i.e., "the sum of the samples plus the "block size".

The FS code is in fact a Huffman code for "some distribution". It provides efficient performance, over a range of entropies from about 1.5 bits/sample up to 2.5 bits/sample.

Backup Option. Another extremely simple code option has been variously called the "backup" option or "default option", originally receiving the designation PSI3. It is defined by

$$PSI3[\tilde{\delta}] \quad \tilde{\delta} \tag{8}$$

General Split-Sample Modes. The concept of Split-Sample Modes (options) as shown in Fig. 4.

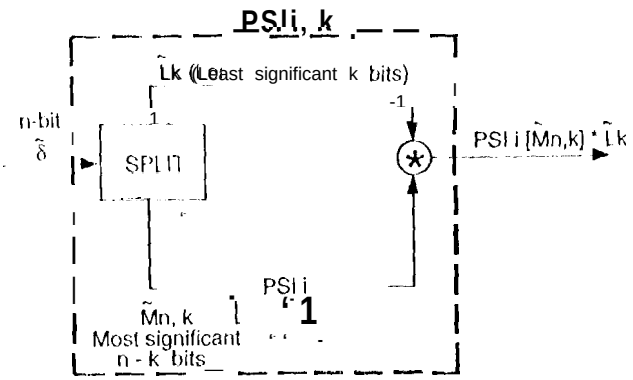

Fig. 4. Split-Sample Mode Structure

As shown, each sample of input $\tilde{\delta}$ is first "split" into their least-significant k bits and their most-significant n-k bits. The least-significant k bits of each sample are combined to form the block $\tilde{L}_k$ and the most significant bits are combined into $\tilde{M}^{n,k}$.

The coding of $\tilde{\delta}$ is completed by coding $\tilde{M}^{n,k}$ with an algorithm named PSIi and then concatenating the result with $\tilde{L}_k$. The overall process is named PSIi,k so that

$$PSIi,k[\tilde{\delta}] = PSIi[\tilde{M}^{n,k}] * \tilde{L}_k \qquad (9)$$

We will use specific numbered coders for PSIi in this paper. But more generally i in Eq. 9 is really a naming label as in Fig. 2.

Simplest Split -Sample Mode, PSI1 ,k. For a specific coder PSIi in Fig. 4, the parameter k defines a set of n+ 1 code options. Various Split-sample modes have been used, starting with an adaptive coder named 1'S14 in place of PSIi back in 1970.[7] But all current interest focusses on the set of code options that are obtained by using PSI1 of Eq. 6 in place of PSIi. this is both because of the extreme simplicity of these opti ons as well as evidence of performance optimality under ideal conditions.

First note the following equivalences:

$$PSI\ ,0 \equiv PSI1 \qquad (10)$$

and

$$PSI\ ,n \equiv PSI3 \qquad (11)$$

so that PSI1 and the Bacl up Option are both special cases of Split-sample Modes.

Pen shu Yeh of Goddard Space Flight Center showed that if the prediction error distribution (see Fig. 2) is Laplacian, then the Split-sample code option 1'S1 1,k is equivalent to a Huffman code with its optimum performance centered at k+ ? bits/sample.[8],[9] Ia practice the Dynamic Range for this code option is approximately -f ().5 bits/sample and centered around k+ 2. Performance for PSI1,k is illustrated in Fig. 5.

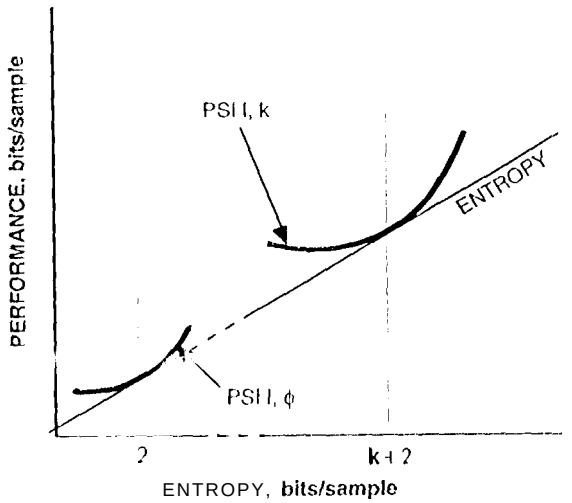

Fig. 5. PSI1,k Performance

Low Entropy Option. Ref. 3 and earlier documents defined a code option called PSI0 that operates efficiently for entropies from about 0.75 bits/sample up to 1.5 bits/sample. Space does not permit detailed discussion here. Also Yeh introduced another option that performs similarly.[1] Later discussions will note approaches for operating at "very" low entropies. For further discussion, note that PSI0 is a Split-sample Mode too

$$PSI0 \equiv PSI0,0 \qquad (1?)$$

Adaptive Coder 1'S114

Ref. 3 defines a set of N-option coders named PSI14 by including the Backup option and N-1 adjacent Split-sample modes from the list in Table 3. The table shows each coder along with their Dynamic Ranges and the parameter λ.

With the selection of Split-sample Modes starting with row λ, the result is an adaptive coder with an extended Dynamic Range (including the cost of identifiers) starting at an entropy of approximately λ + 0.5 and ending at around min{ λ + N - 0.5, n} when $\lambda \geq 1$. The lower end of the Dynamic Range depends on the low-entropy option that is used. For the one used in Ref. 3, this is around 0.75 bits/sample.

Table 3. Options for PSI14

Split-Sample Name	Dynamic Range (bits/sample)	λ
'SI0,0	((0.75, I .5)	0
's1 I,()	(1 .5, 2..5)	1
'SI1,1	(2.5, 3.5)	2
PSI1,k	(k+ 1.5, k+2.5)	k-2
PSI3= *I'sll* **,11**	na	na

The parameter λ provides the means to move a limited Dynamic Range (e.g., suppose N= 4) over an expected region of data entropy.

Identifier Overhead. Assuming a standard fixed-length representation for the code identifier, id (see ID(·) in Fig. 3), an N option PSI14 requires

$$\lceil \log_2 N \rceil / J \quad \text{bits/sample} \qquad (13)$$

in overhead. But this apparent penalty is really only a penalty if the same code option were always selected. But the varying data characteristics of real data sources will typically cause enough switching between code options to more than make up for the identifier's overhead. Otherwise, why build an adaptive coder? However, block size certainly is a consideration. Most versions of PSI14 have been built around a block size of J= 16 where overhead is 3/8 bits/sample when N= 8 and 1/4 bits/sample with N up to 16.

Ref. 3 looks into this issue further, coding sequences of identifiers with another N= 4 option PSI14. Tests showed that in special situations the overall code rate for an N= 16 option PSI14 could be reduced by as much as 0.1 bits/sample.

Assignment of Identifiers. Ref. 3 sets id = N-1 whenever PSI3 is used, whereas recent hardware implementations always use id = 2^n - 1. All other identifier assignments are established by parameter λ which defines the first option in Table 3 that will be used. Basically, the first option used is assigned id = 0, the following option gets id = 1, and so on. See Ref. 3 for more details.

Performance Summary. Fig. 6 roughly summarizes the performance characteristics of PSI14 with parameters N and λ (assuming nominal values for J).

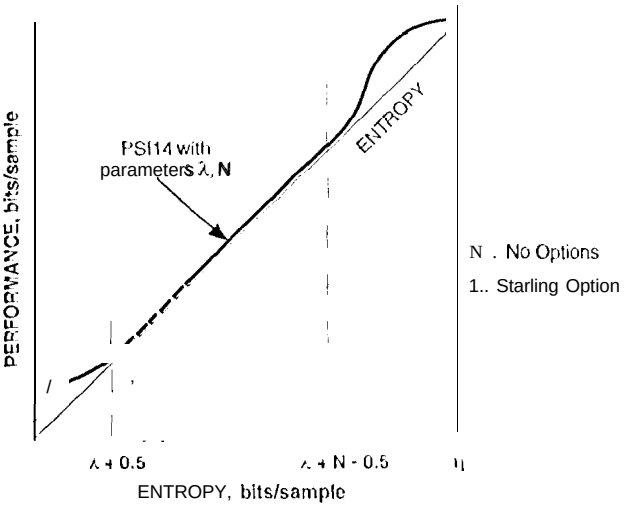

Fig. 6. PSI14 Performance Summary

Implementations. For λ ≥ 1 the ***Huffman equivalent code options that make up a PSI14 really don't require any memory to store codewords***. This simplicity dramatically eases the implementation problem.

The Jet Propulsion Laboratory produced the first VLSI version of PSI14 in 1990. Its parameters were λ= 1, J= 16, n= 12 and N= 11. The decision mechanism for determining which of the 11 options to use, was based solely on the length of the Fundamental Sequence as generated by PSI1 (see Eq. 7). Space qualified improvements to this first chip will be used to support imaging and another instrument on the billion dollar Cassini mission.

A more generic VLSI coder and decoder were developed in 1991 at the Microelectronics Research Center, formerly at the University of Idaho and now located at the University of New Mexico, under the direction of Goddard's Warner Miller and Pen-shu Yeh.[9],[10] The PSI14 parameters were λ= 1, J=16, n= 4,5,...,14 and N=12. The decision mechanism for this coder used calculations of the actual code rates for each option.

Laboratory tests of these 1.2 μm devices showed coding rates at up to 900 Mbits/sec. Following a technology transfer, these chips have been sold commercially by Advanced Hardware Architectures in Pulman, Washington.

A software version of PSI14 (and more) by Alan Schlutsmeyer is currently being used to represent the data of six instruments aboard the Galileo spacecraft now orbiting Jupiter.[11]

On the subject of Galileo, an earlier adaptive coder that uses PSI0, PSI1, PSI3, an option called PSI2 and Split-sample modes incorporating PSI2, is the primary engine of a "lossy" rate controlled hardware image compressor.[12] This compressor returned the recent Ganymede pictures that have been widely shown in the newspapers.

Coder PSI14,K

Look again at the Split-sample Mode structure in *Fig. 4*. In developing PSI14 we have been utilizing the code option PSI1 in place of the general PSIi. But this is not necessary. Instead, insert adaptive coder PSI14 in place of PSIi. Then the split-sample mode PSIi,k becomes PSI14,K (where we now capitalize the K to help keep the distinction).

The difference now is that instead of an internal coder with a short Dynamic Range centered around 2 bits/sample (PSI1), the internal coder is one with a broad and adjustable Dynamic Range (via λ). If K=0, then PSI14,0 ≡ PSI14. As K is **incremented** by 1, the "effective" Dynamic Range for the internal PSI14 is moved **upward** by 1 bit/sample.

Thus K acts much like λ. But adjusting K can be used, after the fact, to broaden the utility of an existing limited PSI14 (e.g., an 8-option coder with λ= 1 working on 16 bit data exhibiting entropies ranging from 5 to 12 bits/sample).

Splitting Before the Preprocessor. Another amazing result: The splitting of samples for Split-sample Modes can be performed ***before or after preprocessing*** (as described here) with almost identical performance results. This can be a very useful observation to keep in mind (i.e., think PSI14,K) particularly when making use of an existing but limited PSI14. **QUIZ:** How do you use "a hardware PSI14 coder and basic preprocessor (as in Fig. 2), which is designed to work on image data quantized to at most n = 12 bits/sample," to represent lines of image data quantized to 14 bits/sample?

Some Comparisons

Today's best alternatives to this collection of coding techniques, now widely known as the Rice Algorithms, include LZW (for inventors Lempel, Ziv and Welch), Adaptive Huffman and Arithmetic Coding. Most computer disk compression programs are based on LZW. Adaptive Huffman (a HUF) is an algorithm that constantly updates the code being used. Arithmetic Coding (ARITH) is a sophisticated approach often associated with the International Business Machines Corporation.

Jack Venbrux compared the performance of these algorithms with a version of PSI14 on a broad range of image data. In these comparisons, each compression technique used the same Rice Algorithm preprocessing.[13] The results are repeated here for convenience in Fig. 7 and speak for themselves. Later papers by Venbrux and Yeh also showed a significant advantage over the lossless mode of JPEG, the commercial standard for image compression.[9]

Other Rice Algorithms

Refs. 14 and 15 described a globally adaptive, rate controlled, lossy image compression system called RM2. To assist in coding the RM2 transform coefficients, special techniques were developed and finally described in detail in Refs. 6 and 16. The adaptive techniques (which included code options PSI9 and PSI10) provided efficient representation for very low entropy, non-stationary, memoryless sources, including a specific method for coding binary memoryless sources with changing statistic. The latter algorithms were also used to achieve a 20:1 average compression ratio for the Gamma Ray Spectrometer on the ill-fated Mars Observer spacecraft (and will be used on its replacement).[17]

About the CCSDS Standards

The international CCSDS standards recommendations for lossless data compression is now nearing completion in its "red book" phase primarily due to the efforts of Warner Miller and Pen-shu Yeh of Goddard Space Flight Center.[1]

The overall coder specification is essentially a modified version of PSI14 described above, with fixed λ. Yeh has incorporated an alternate to the PSI0 defined in earlier references that codes the 2nd extension of incoming preprocessed data. Additionally, she has introduced a run-length code option that is very effective in taking advantage of the occurrence of blocks that are ***all zero***, something that often happens when data is processed.

The lossless coding Red Book provides specifications of these new features and other subtleties, as well as details to ensure compatibility with packet telemetry standards.

Second generation VLSI coders and decoders that support much of the standards specification have been under development at the Microelectronics Research Laboratory, University of New Mexico, by Jack Venbrux.

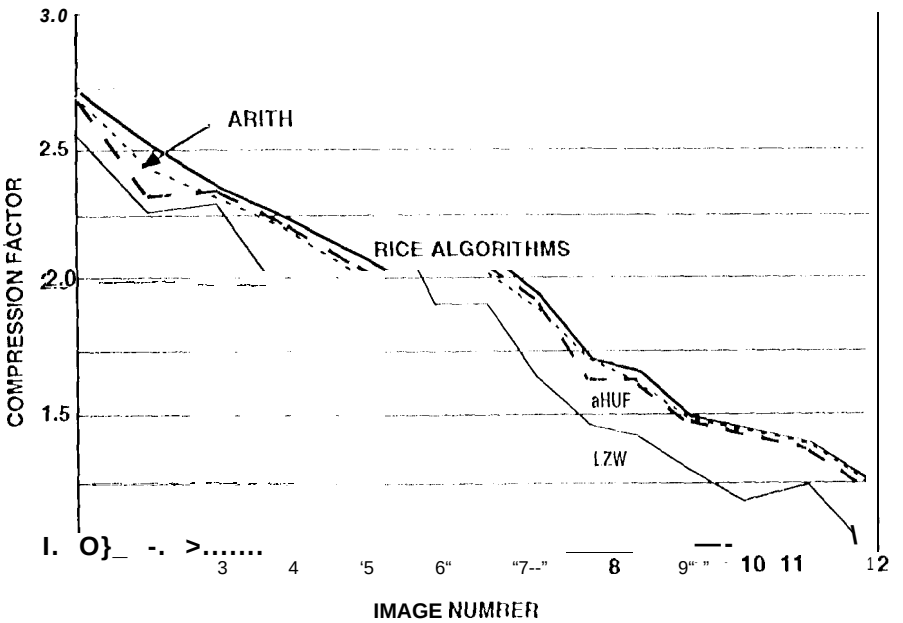

Fig. 7. Performance Comparisons

Acknowledgments

The author would like to offer thanks to key individuals and organizations whose contributions and support have helped achieve the results described herein. To Alan Schlutsmeyer for his extraordinary programming talents. To Warner Miller, Pen-shu Yeh, Jack Vendrux and Jun-ji Lee for their technical and moral support. To Gamo Publications, Burbank, California for their always professional publication support.

Some of the research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and the University of California at Los Angeles, under contracts with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, the Jet Propulsion Laboratory, California Institute of Technology.

References

1) "Recommendation for Space Data System Standards: Lossless Data Compression," CCSDS 121.0-R-2, **Red Book**, August 1996.

2) Robert F. Rice, "Data Compression and Error-Protection Coding," **Proceedings of the 32nd Space Congress**, Cocoa Beach, Florida, USA, April 25-28, 1995.

3) Robert F. Rice, "Practical Universal Noiseless Coding Techniques, Part III, " **JPL Publication 91-3**, Jet Propulsion Laboratory, Pasadena, California, USA, Nov. 15, 1991.

4) D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," **Proc. IRE**, Vol. **40,** pp. **1098-1101,** 1952.

5) **R.** F. Rice and J. Lee, "Some Practical Universal Noiseless Coding Techniques Part II, " **JPL Publication 83-17**, Jet Propulsion Laboratory, Pasadena, California, March 1, 1983.

6) R. F. Rice, "Some Practical Universal Noiseless Coding Techniques, " **JPL Publication 79-22,** Jet Propulsion Laboratory, Pasadena, California, March 15, 1979.

7) R. F. Rice- and J. R. Plaunt, "Adaptive Variable Length Coding for Efficient Compression of Spacecraft Television Data," **IEEE Trans. on Communication Technology**, Vol. COM- **19,** Part **1,** Dec. **19-/1,** pp. 889-897.

8) Pen-shu Yeh, et al., "On the Optimality of code Options for a Universal Noiseless Coder," **JPL**

Publicat ion 91-2, Jet Propulsion Laboratory, Pasadena, California, February 15, 1991.

9) Pen-shu Yell et al., "On the Optimal ity of a Universal Lossless Coder," **Proceedings of the AIAA Computing in Aerospace 9 Con - ference, Sal)** Diego, California, October **19-21,** 1993.

10) Jack Venbrux, et al., "A Very High Speed Compression/Decompression Chip Set," **Proceedings IEEE Data Compression Conference,** Snowbird, Utah, April 1911; and J P l, **Publication 91-13,** Jet Propulsion Laboratory, Pasadena, California, July 15, 1991.

I 1) Robert Rice, Robert Mehlman, "1 .ossless Compression and Rate Control for the Galileo NJMS Instrument," **Proceedings of the Eighth Annual AIAA/USU Conf. on** small **Satellites, 1** .ogan Utah, USA, Sept. 1, 1994.

1?) R.F. Rice, et al., "Block Adaptive Rate Controlled Image Data Compression," **Proceedings of 1979 Nat. Telecommunications Conference,** Washington, 1)('., Nov. 1979.

13) Jack Venbrux, et al., " A Very High Speed Compression/Decompression Chip Set, " **IEEE Transactions on Circuits and Systems,** Vol. 2, No. 4, Dec. 1992.

14) R. F. Rice, "An Advanced Imaging Communication System for Planetary Exploration," **Vol. 66 SPIE Seminar Proceedings,** Aug. **21-2?, 1975,** po. 70-89.

15) Robert Rice, "End-to-End Information Rate Advantages of Various Alternative Communication Systems, " **JPL Publication 82-61,** Jet Propulsion Laboratory, Pasadena, California, September 1, 1982.

16) R. F. Rice, "Practical Universal Noiseless Coding," **SPIE Symposium Proceedings, Vol. 207,** San Diego, California, Aug. 1979.

17) R. F. Rice, J. Lee, "Noiseless Coding for the Gamma Ray Spectrometer," **JPL Publication 85-53,** Jet Propulsion Laboratory, Pasadena, California, June 1985.